A Holistic Approach to OLAP Sessions Composition: The Falseto Experience

Julien Aligon, Kamal Boulil, Patrick Marcel, Verónika Peralta

A holistic approach to OLAP sessions composition: The Falseto experience

Julien Aligon, Kamal Boulil, Patrick Marcel, Verónika Peralta
University of Tours
Blois, France
firstname.lastname@univ-tours.fr

July 10, 2015

Abstract

OLAP is the main paradigm for flexible and effective exploration of multidimensional cubes in data warehouses. During an OLAP session the user analyzes the results of a query and determines a new query that will give her a better understanding of information. Given the huge size of the data space, this exploration process is often tedious and may leave the user disoriented and frustrated.

This paper presents an OLAP tool named Falseto (Former AnalyticaL Session for lEss Tedious Olap), that is meant to assist query and session composition, by letting the user summarize, browse, query, and reuse former analytical sessions. Falseto's implementation on top of a formal framework is detailed. We also report the experiments we run to obtain and analyze real OLAP sessions and assess Falseto with them. Finally, we discuss how Falseto can be seen as a starting point for bridging OLAP with exploratory search, a search paradigm centered on the user and the evolution of her knowledge.

1 Introduction

While it is universally recognized that OLAP tools have a key role in supporting flexible and effective exploration of multidimensional cubes in data warehouses, it is also commonly agreed that the huge number of possible aggregations and selections that can be operated on data may make the user experience disorientating. OLAP queries are usually formulated in the form of sequences called OLAP sessions. During an OLAP session the user analyzes the results of a query and, depending on the specific data she sees, determines a new query that will give her a better understanding of information. This new query is often constructed by modifying the previous query, applying simple primitives

like rollup, drill down or slice/dice. Noticeably, given the huge number of such modification possibilities, this exploration process is often seen as tedious and may leave the user disoriented and frustrated.

To make the user experience less disorientating, it has recently been suggested that database systems could take advantage of query logs [14]. Logs can indeed be used for learning user preferences [12], for query recommendation [7, 21], for query auto-completion [15], or for query composition [16]. Similar efforts were also done in the context of OLAP systems [4, 9, 2], but so far no usable OLAP tool implements them, and this leds us to the development of Falseto.

Falseto (Former AnalyticaL Session for lEss Tedious Olap) aims at assisting the user analyzing data cubes, by letting her summarize, browse, query, and reuse former analytical sessions. In particular, Falseto's recommender system automatically suggests to the user an analytical session considered as relevant regarding her current session. Falseto also lets the user browse former sessions, which is particularly useful when a user initiates her session and the recommender system cannot suggest queries since no current session exists. Since the set of former sessions can be very large, Falseto presents first summarized versions of the sessions in a way that makes it easy to navigate and query them.

The paper is structured as follows. Section 2 introduces Falseto's functionalities by means of examples. Section 3 describes the logical framework underlying Falseto and Section 4 presents its implementation. Section 5 reports the lessons learned through various tests aiming at obtaining, analyzing and assessing Falseto with real OLAP sessions. After briefly reviewing related work, we discuss how Faseto can be a starting point of future research to bridge the gap between OLAP and exploratory search in Section 6. Section 7 concludes the paper.

2 Getting familiar with Falseto

Assume you are an OLAP user who has to analyze census data obtained from the IPUMS (Integrated Public Use Microdata Series) website [18], and gathered under the form of a multidimensional cube whose schema is depicted Figure 1. It has 5 hierarchies describing people sex, race, occupation and residence place, as well as census year. They are organized as follows:

- Sex is a one-level hierarchy, containing two values: Male and Female.
- Race is a three-level hierarchy. MRN (stands for Major Races Number) indicates the number of major races mix of a person (e.g. 2). Racegroup group races in major categories (e.g. Black and Asian) and Race lists detailed races (e.g. Black and Korean).
- Occupation is a four-level hierarchy. Category indicates 9 major occupation categories, that are specialized in Sub-caterories and Branches. Finally, Occupation lists detailed occupations.

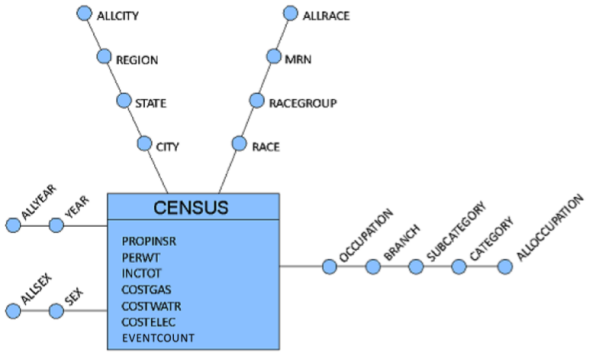

Figure 1: IPUMS multidimensional schema

- City is a three-level hierarchy describing people residence place. Region indicates 10 United States regions. State indicates state codes (e.g. CA for California) and City lists United States cities.
- Year is a one-level hierarchy, containing six values, from 2000 to 2005.
- The schema describes six census indicators: total personal income (INCTOT), annual property insurance cost (PROPINSR), annual gas cost (COSTGAS), annual water cost (COSTWATR), annual electricity cost (COSTELEC) and person weight (PERWT). Four measures are defined for each indicator according to 4 aggregation functions (sum, max, min and average), for example SUMINCTOT, MAXINCTOT, MININCTOT and AVGINCTOT. A 25th measure (EVENTCOUNT) counts the number of facts.

More specifically, assume that you have to answer the following, rather vague, analytical question: What is the profile of female workers that better resists to a drop of average income? You would probably try to navigate the IPUMS cube, playing with aggregation levels and filters, to discover this profile.

Falseto, whose main user interface is depicted in Figure 2, can be used to graphically create the queries of your session, execute them, view the result and decide on what the next query should be. Designing queries is done in the query design tab (bottom right corner of Figure 2). The interface refers to the Dimension Fact Model [10] and allows to add each component of the query i.e the measure set, the group-by set and the selection set. The reason for choosing this way of representing OLAP queries is, aside the popularity of the DFM, to have a simple, uniform, non-textual model to depict the queries using the schema of the cube, which allows to focus on the three components of the query. Falseto also enables to devise a sequence of OLAP queries, called OLAP session, in order to answer your analytical goal. Each time a query is created and evaluated, it is recorded and placed to the Session Viewer panel

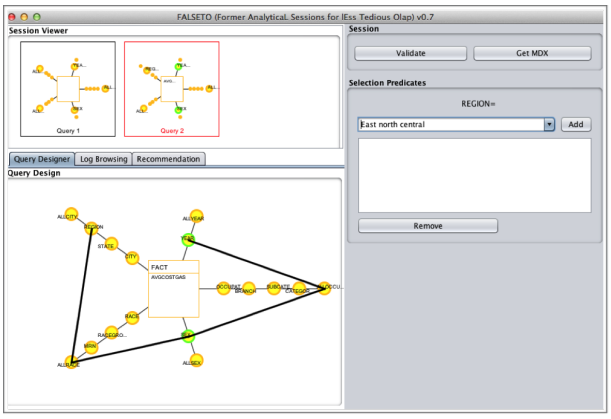

Figure 2: Falseto's main user interface

(top left corner of Figure 2). This is the viewer for the past queries of your session. Each query there can be subsequently retrieved, zoomed, reevaluated, edited, and used as a parameter for the holistic functionalities of Falseto.

The advanced features of Falsteo are called holistic functionalities since they consider sessions in their entirety and do not treat queries independently. Of course, all queries retrieved by means of these functionalities can be reused and edited.

The first holistic functionality allows to browse the past sessions recorded in the query log. The log browsing tab of Falseto provides an overview of these past sessions (see Figure 3). Each of the different panels represents a cluster (i.e. a subset) of past sessions that are considered similar. The text in each panel indicates the fragments that appear in all the sessions (group by levels are in yellow, selections in green and measures in black). A left-click on a cluster displays a summary of the cluster under the form of a sequence of queries (bottom of Figure 3). Each query of this session summarizes a portion of each session of the cluster by displaying the common measures and selections, and the coarsest group-by set of the portion.

It is also possible to scroll down the detailed view in order to display each past session that compose the cluster. Another possibility is to split a cluster into several clusters of shorter sizes, or to fusion clusters to have a coarser view of the log. Because the browsing of each cluster or individual session can also be tedious Falseto also offers filtering operators to reduce the number of clusters and sessions to browse. The specialization-based search retrieves from the set of past sessions only those that include at least one query containing the selections, measures and finer (or identical) group by sets of the query given as the input of the operator. The similarity-based search retrieves from the set of past sessions only those that include at least one query that is similar to the query given as input of the operator. These two operations can also be used by giving not a

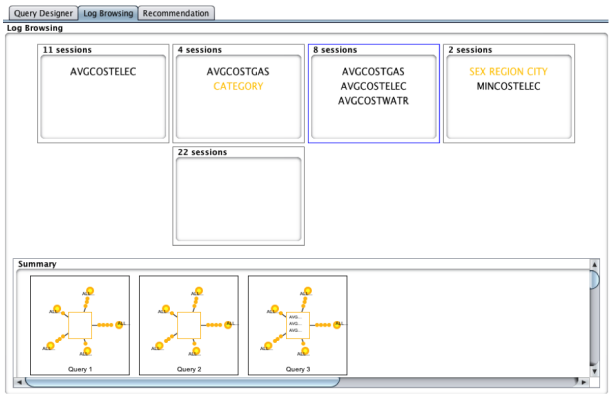

Figure 3: Log browsing with Falseto

query but a session as input.

Finally, Falseto implements a recommender system that suggests a sequence of queries based on what you have done in your current session. This suggestion comes from a past session (recorded in the log) whose beginning is found similar to your session, and can be seen as a potential, customized future for your current session. This recommender system can be seen as an instantiation of the generic framework proposed in [8, 9] with some of the similarity measures described in [5]. More details are given in Section 4 and the full description can be found in [3, 2].

Note that the examples and figures used in this section come from the Falseto tutorial, available online on the Falseto web site (http://vega.info.univ-tours.fr:29082/tea/index.jsp) Falseto itself can be downloaded from this site.

3 The logical framework

The logical framework underlying Falseto relies on the separation between queries and logs. The query design functionnalities of Falseto rely on a simple query model and language described Section 3.1, while holistic functionnalities rely on a log manipulation language described Section 3.2.

3.1 The query edition language

Query edition is done with a simple language for manipulating dimensional queries [10]. We now introduce the data model, the query model and finally the operators of this language. To keep the formalism simple, we consider cubes under a ROLAP perspective, described by a star schema [17]. More precisely, we consider that a dimension consists of one hierarchy, and we consider simple hierarchies without branches, i.e., consisting of chains of levels.

Let $\mathcal{L}$ be a set of attributes called levels, and for $L \in \mathcal{L}$, a member is an element of $Dom(L)$. A hierarchy h_i is a set $Lev(h_i) = \{L_0, \ldots, L_d\}$ of levels together with a *roll-up* total order $\succeq_{h_i}$ of $Lev(h_i)$, which is such that, for any L_j and L_k in $Lev(h_i)$, $L_k \succeq_{h_i} L_j$ if $Rollup(L_j) = L_k$.

A *multidimensional schema* (or, briefly, a *schema*) is a triple $\mathcal{M} = \langle A, H, M \rangle$ where:

- A is a finite set of *levels*, whose domains are assumed pairwise disjoint,
- $H = \{h_1, \ldots, h_n\}$ is a finite set of *hierarchies*, (such that the $Lev(h_i)$'s for $i \in \{1, \ldots, n\}$ define a partition of A);
- a finite set of *measure* attributes M, each defined on a numerical domain $Dom(m)$.

Given a schema $\mathcal{M} = \langle A, H, M \rangle$, let $Dom(H) = Lev(h_1) \times \ldots \times Lev(h_n)$; each $G \in Dom(H)$ is called a *group-by set* of $\mathcal{M}$.

Definition 3.1 (Fragment-based OLAP query) *A* query *over schema $\mathcal{M} = \langle L, H, M \rangle$ is a triple $q = \langle G, P, Meas \rangle$ where:*

1. *$G \in Dom(H)$ is the query group-by set;*
2. *$P = \{p_1, \ldots, p_n\}$ is a set of Boolean predicates, one for each hierarchy, whose conjunction defines the* selection predicate *for q; they are of the form $l = v$, or $l \in V$, with l a level, v a value, V a set of values. Conventionally, we note $p_i = TRUE_i$ if no selection on h_i is made in q (all values being selected);*
3. *$Meas \subseteq M$ is the measure set whose values are returned by q.*

Definition 3.2 (OLAP Session and log) *Let $\mathcal{M} = \langle A, H, M \rangle$ be a multidimensional schema and S_C be a set of queries over $\mathcal{M}$ in a given language. A session s of k queries $s = \langle q_1, \ldots, q_k \rangle$ over $\mathcal{M}$ is a function from an ordered set $pos(s)$ of integers (called positions) of size k to S_C. A log L is a finite set of sessions, noted $L = \{s_1, \ldots, s_p\}$.*

The operators of the query edition language allow to derive a query expression from another one by affecting a particular component of the query (group-by set, selection set or measure set). The operators are similar to those traditionally used in OLAP (such as roll-up, drill-down, or slice-and-dice) but instead of being applied on cubes, they are applied on queries.

The first two operators allow to modify the group-by set g, by changing a level, for a given hierarchy h_i, to a coarser or finer granularity level. Two operators modify the selection set P. The first operator adds a member c'_i of a level l'_i to a set of members X_i of the same level as l'_i of a predicate $p_i \in P$, for a given hierarchy h_i. The second operator removes a member c'_i of a level l'_i to a set of members X_i of the same level as l'_i of a predicate $p_i \in P$, for a given hierarchy h_i. Finally, two operators allow modifying the measure set $Meas$, by either adding or removing a measure fragment.

Definition 3.3 (Query edition operators) *Let $q = \langle g, P, Meas \rangle$ be a query, h_i be a hierarchy, $l_i \in \{c'_i\}$ be a selection predicate and m a measure.*

- $Rollup_{int}(q, h_i) = \langle g', P, Meas \rangle$ *where* $l_i \in Lev(h_i)$ *and* $g' =< l_1, \ldots, Rollup(l_i), \ldots, l_n >$, *if* $l_i \neq ALL_i$, *undefined otherwise.*
- $Drilldown_{int}(q, h_i) = \langle g', P, Meas \rangle$ *where* $l_i \in Lev(h_i)$ *and* $g' =< l_1, \ldots, Drilldown(l_i), \ldots, l_n >$, *if* $l_i \neq DIM_i$, *undefined otherwise.*
- $AddSelection(q, l_i \in \{c'_i\}) = \langle g, P', Meas \rangle$ *where* $p_i \in P'$ *with* $p_i = l_i \in X'_i$ *such as* $X'_i = X_i \cup \{c'_i\}$
- $RemoveSelection(q, l_i \in \{c'_i\}) = \langle g, P', Meas \rangle$ *where* $p_i \in P'$ *with* $p_i = l_i \in X'_i$ *such as* $X'_i = X_i - \{c_i\}$
- $AddMeasure(q, m) = \langle g, P, Meas' \rangle$ *where* $Meas' = Meas \cup \{m\}$.
- $RemoveMeasure(q, m) = \langle g, P, Meas' \rangle$ *where* $Meas' = Meas - \{m\}$.

3.2 The log manipulation language

The holistic functionalities of Falseto can be easily described by expressions in a language that manipulate logs, i.e., sets of sessions [6]. This algebraic language features 5 operations. For the sake of simplicity, the names and symbols of the operations are the same as the ones used in the relational algebra. Two of them are unary operations: selection σ and group by and aggregation π. Three of them are binary: join $\bowtie$, union $\cup$ and difference $\setminus$.

The selection operation enables to select from a log those sessions satisfying a given condition. This condition is given under the form of a relation to another session.

Definition 3.4 (Selection) *Let L be a log, s be a session and θ be a binary relation over sessions.*

$$\sigma_{\theta,s}(L) = \{s' \in L | \theta(s, s')\}$$

As logs are defined as sets of sessions, the set union and set difference operations can be used to manipulate logs. The definitions are straightforward:

Definition 3.5 (Set operations) *Let L and L' be two logs.*

$$L \cup L' = \{s \in L \text{ or } s \in L'\} \text{ and } L \setminus L' = \{s \in L | s \notin L'\}.$$

The join operation enables to combine the sessions in two logs, that are related through a relation θ.

Definition 3.6 (Join) *Let L and L' be logs, f be a binary functions outputting a session and θ be a binary relation over sessions.*

$$L \bowtie_{\theta,f} L' = \{f(s, s') | s \in L, s' \in L', \theta(s, s')\}.$$

Note that intersection can be simulated with the join operation. Indeed, $L \cap L' = L \bowtie_{same,first} L'$ with, for any two sessions s, s', $first(s, s') = s$ and $same(s, s') \equiv (s = s')$.

Note also that this join operation is not symmetric unless θ is symmetric and $f(s, s') = f(s', s)$ for all s, s'. Moreover, the f function is needed for closeness reason. In that sense, it is inspired by the join operation defined in [1] for joining cubes.

This operator allows to group sessions that are related to one another through relation θ, and to aggregate them using an aggregation function agg.

Definition 3.7 (Grouping and aggregation) *Let L be a log, agg be a function aggregating a set of sessions into a session, and θ be a binary relation over sessions.*

$$\pi_{\theta,agg}(L) = \{agg(\{s' \in L | \theta(s, s')\}) | s \in L\}.$$

We briefly review the properties of the language in terms of closeness, completeness and minimality. It can easily be seen that the language is closed under composition, the result of any operation being a set of sessions. The language is complete. Indeed, for any pair of logs L and L', there is an expression that enables to transform L into L'. The transformation would proceed as follows:

1. pick a session s in L with σ,
2. transform it into a session s' of L' using π,
3. repeat the previous steps until all sessions of L' are formed,
4. union all transformed sessions to form L'.

The language is not minimal since σ can be simulated with $\bowtie$. Indeed, $\sigma_{\theta,s}(L) = L' \bowtie_{\theta,f} L$ where $L' = \{s\}$ and $f(s, s') = s'$. It can easily be seen that removing σ makes the language minimal.

4 Under the hood of Falseto

Falseto is implemented as a standalone Java application, on top of the Mondrian OLAP engine, and uses the Weka machine learning library. Falseto can be used with any database system compatible with Mondrian and can be used to query any cube with a star schema.

In terms of query edition, Falseto can be used to to constructed a query on a given schema from scratch, edit the query using the query edition operators, and evaluate the query over an instance of the schema. All evaluated queries are kept in an in-memory structure representing the current session of the user, who can go back to them for edition or reevaluation purpose. The sessions of the log, that can be retrieved using the holistic functionalities, are loaded in memory when Falseto is started, and their queries can also be edited and evaluated.

The holistic functionalities of Falseto can be described by expressions in the log manipulation language introduced above. Noticeably, important inputs of

these expressions are the relations over sessions that are used to parametrize the operators. This is why, before giving the expressions corresponding to the holistic functionalities, we start by introducing the relations between queries and sessions that will be used as the parameters of the operators.

4.1 Relations over queries and sessions

To describe the holistic functionalities of Falseto, we describe two types of relations. The first type are specialization relations, enabling the descriptions of groups of queries or sessions at various levels of details. The second type are similarity relations, allowing the description of how close or distant two queries, or two sessions, are. We start by the specialization relations.

Considering two group-by sets G and G', recall that we note $G \succeq_H G'$ if G is more general than G' in the sense of the group-by lattice, whose top element $G^\top$ is the coarsest group by set. For two sets of selection predicates P, P', we note $P \succeq_p P'$ if P is more selective than P', i.e., $val(P) \subseteq val(P')$, where $val(X)$ is the set of values used in predicate X. Finally, for two measure sets M, M', we note $M \succeq_m M'$ if $M \subseteq M'$.

Definition 4.1 (Query specialization) *Let $q = \langle G, P, M\rangle$ and $q' = \langle G', P', M'\rangle$ be two queries over the same schema. q is more general than q', noted $q \succeq q'$, if $G \succeq_H G'$ and $P \succeq_p P'$ and $M \succeq_m M'$.*

We introduce a specialization relation over sessions that is based on a specialization relation over queries. In what follows, $s \succeq s'$ denotes that a session s is more general than a session s'.

Definition 4.2 (Session specialization) *A session $s = \langle q_1, \ldots, q_{n_s}\rangle$ is more general than another session $s' = \langle q'_1, \ldots, q'_{n_{s'}}\rangle$, written $s \succeq s'$, if there exists a sequence of $n_{s'}$ integers $\{i_1, \ldots, i_{n_{s'}}\}$ with $i_1 = 1$, $i_{n_{s'}} = n_s$ and, for $k \in \{1, n_{s'} - 1\}$, $i_k \leq i_{k+1}$, such that, for all $j \in \{1, \ldots, n_{s'}\}$, it is $q_{i_j} \succeq q'_j$.*

The intuition of this specialization relation is given Figure 4. Note that if $s \succeq s'$ then the sequence of integers $\{i_1, \ldots, i_{n_{s'}}\}$ defines a partition $P = \{p_1, \ldots, p_{n_s}\}$ of $queries(s')$ where the p_i's can be ordered according to the queries of s, and the queries in each p_i, each less general than q_i, constitute a sub-session of s'.

Finally, we suppose a function $dist$ applied to pairs of sessions, giving the distance between two sessions. Similarity relations over sessions can be defined from such a function. For instance, $sim(s, s') \equiv dist(s, s') \leq \alpha$ for some real α indicates that two sessions s and s' are similar if their distance is below a threshold. Falseto implements such a relation, that is based on the Smith-Waterman alignement algorithm, whose goal is to efficiently find the best alignment between subsequences of two given sequences by ignoring their non-matching parts. An extension of this algorithm was proven a good measurement of OLAP session similarity in [5].

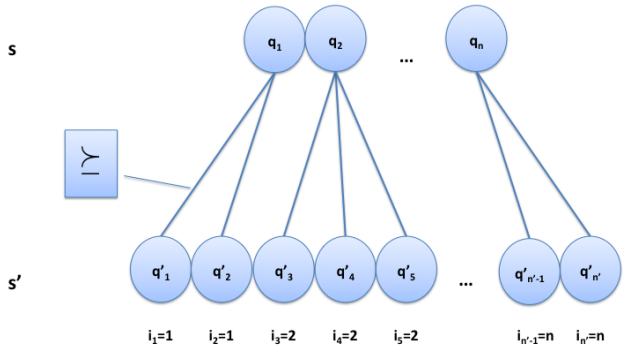

Figure 4: Specialization relation over sessions

4.2 Holistic functionalities

Recall from Section 2 that Falseto includes three holistic functionalities: log searching, log browsing and session recommendation.

Log search To search the log for useful sessions, Falseto implements different versions of the log selection $\sigma_{\theta,s}$. A first version enables the filtering of a log using for θ the sim relation based on the extension of the Smith-Waterman algorithm. This operation returns those sessions of the log that are close to the session s given as the second parameter. Likewise, a log can be filtered using for θ the specialization relation over sessions defined above. This version of the selection is very useful as it allows to search a log for queries by giving only a partial description of these queries. For instance, to search the log L for sessions containing queries with selection $year = 2013$, it is sufficient to use $\sigma_{\trianglerighteq,\langle\langle G_{All},P=\{year=2013\},Meas\rangle\rangle}(L)$, where G_{All} is the coarsest group-by set and $Meas$ is the set of all measures.

Log summarization and browsing To allow for a user-friendly browsing of the log, Falseto implements operations for summarizing the log, using the π grouping and aggregation operator. The basic summarization operation allows to form pairs of most similar sessions, that are subsequently aggregated using the specialization relation. The θ relation used for grouping corresponds to $\theta_{sim}(s, s') \equiv \nexists s'' \in L$ such that $sim(s, s'') > sim(s, s')$ and $sim(s', s'') > sim(s, s')$. The aggregation function merges pairs of sessions into a session more general than the two elements of the pair. It uses a tilted time window [11] to ensure that the same portion of the two sessions, potentially

of different sizes, are merged. This window is logarithmic and gives priority to recent queries of the sessions. It can be declaratively written as follows: $agg_{merge}(s, s') = \{s''$ such that both $s'' \succeq s$ and $s'' \succeq s'$ and for all $i \in [1, |s''|], s''$ is the least session w.r.t. $\succeq$ such that both $\langle q''_i \rangle \succeq s|i$ and $\langle q''_i \rangle \succeq s'|i\}$ where the q''_i's are the queries of s'' and $s|i$ represents the portion of a session s that is corresponds to q_i in the sense of the tilted time window.

The π^* operation is a generalization of the basic summarization operation, and summarizes a log so as to obtain only one session generalizing all the sessions in the log. It is implemented as an ascendant hierarchical clustering that at each steps uses $\pi_{\theta_{sim}, agg_{merge}}$ to group pairs of sessions. When Falseto is started, it computes the clustering of the log, stores the resulting dendogram and displays in its log viewer a summary of the log. This summary is not the session that summarizes the complete log since such a session would most likely be too much general to be useful. Instead, Falseto proposes nodes of the dendogram that correspond to summarized sessions that contains at least one query that is not the most general one. To save space on the viewer, these nodes are represented as tag clouds, where the tags are the query fragments. On demand, the user can see the details of the corresponding cluster. The dendogram can also be browsed: each cluster displayed on the viewer can be split to display its two direct descendants. This "drill down" browsing operation exploits the results of a π^* operation, in the sense that, from any session s of the dendogram resulting of π^*, the pair of sessions summarized with agg_{merge} to obtain s is given.

Recommender system Finally, the Falseto recommender system [3, 2] can be described by the three following phases:

1. the *Alignment* phase selects from the log L a set of sessions that are similar to the current session s_{cur} and determines from them a set F of potential futures for s_{cur}. This is implemented using a join operation $F = L \bowtie_{sim, future} L$ where sim is the similarity relation and $future$ is the function that determines in s' the potential future for s.

2. The *Ranking* phase chooses a *base recommendation* r as a subsession of a candidate recommendation in F by considering both its similarity with s_{cur} and its frequency in L. It is implemented by $\{r\} = \pi_{all, agg_{rank}}(F)$ where $all(s, s')$ holds for any s, s' of F and agg_{rank} is the function that compares pairwise the subsessions of F and returns the most relevant one.

3. The *Fitting* phase adapts r to s_{cur} by looking for relevant patterns in the query fragments of s_{cur} and r and using them to make changes to the queries in r, so as to deliver a recommendation r'. It is implemented by a join $r' = \{s_{cur}\} \bowtie_{all, fit} \{l\}$ where l is the log session from which r is extracted and fit is the function used for adapting r to s_{cur}.

We close this section by noting that the holistic functionalities of Falseto, and therefore the specialization relations and similarity relations so far used in Falseto, are exclusively based on the syntax of queries. This is done for the sake

of achieving an interactive response time. Indeed, query answer could have been used instead of query syntax, for instance in the recommender system or the log summarization facilities, but this would have had an substantial impact on the time taken to compute the recommendation or summarize the log. Incorporating query answer in the holistic functionalities of Falseto is part of our future works.

5 Experiments

We tested our approach by running some experiences with OLAP users analyzing the cube whose schema is described in Section 2. We first describe the test protocol and then report the lessons learned.

Our objectives for running these tests was to obtain real OLAP sessions and automatically assess Falseto's holistic functionalities using these sessions. Due to the limited time we had with the users, the goal was not to ask the users to directly assess Falseto's holistic functionalities. Such assessment is part of our future works as explained below.

5.1 Test protocol

The test consisted of developing sessions for 4 analytical questions on the IPUMS cube:

1. (Q1) Is there a trend in the evolution of the average cost of gas for some profiles?
2. (Q2) Are energy costs following the evolution of the average income for some profiles?
3. (Q3) What are the individual profiles to target for a campaign of energy cost reduction (i.e., those that pay too much)?
4. (Q4) Where is it better to live in terms of energy costs, for an individual profile?

Note that these questionnaires can be found online via the Falseto web page (see Section 2).

The 17 OLAP users engaged in the test were students of Master's programs specialized in BI: 7 of them were students of the European Erasmus Mundus IT4BI program (http://it4bi.univ-tours.fr/) and 10 were students of the French program in Information systems and BI of the University of Tours. The test was not part of the program, was not graded and all the participants were volunteers.

Students were first shown the Falseto tool (mostly the query design functionalities) and then they had 2 hours to develop sessions. Then they were asked to identify in the sessions relevant queries, in the sense that the result of these relevant queries should be included in a report summarizing the session. They were also asked to justify the relevance of these queries.

With this corpus of annotated sessions, we tested Falseto's holistic functionalities as follows. We developed different naive algorithms using Falseto's holistic functionalities to automatically generate sessions, and we compared these synthetic sessions with the real ones. More precisely:

1. To understand Falseto's log search capacities, we generated sessions for an analytical question by first randomly choosing a query among the first queries of the log sessions developed for this analytical question. Then, with an increasing probability for ending the session, at each step the current query is used to filter the log and retrieve among the resulting session the query that is the closest to the current one. This query is the new current query (and is removed from the log). For these generated sessions, we report the recall and precision compared to the log queries, computed as follows. Recall is the number of queries of the generated session that are queries developed for the analytical question over the total number of queries developed for the analytical question found in the log. Precision is the number of queries developed for the analytical question over the total number of queries in the generated session. We also measure recall and precision w.r.t. relevant queries. In this case, Recall is the number of relevant queries in the generated session over the number of relevant queries in the log sessions for the analytical question, while precision is the number of relevant queries in the session over the number of queries in the session.

2. To understand Falseto's capacities of suggesting new information when answering an analytical need, we generated sessions by picking a session in the log, removing the final part of the session and replacing it with the session suggested by Falseto's recommender system. For this we report the session novelty (the number of queries that cannot be found in the log), similarity of the synthetic session to the sessions developed for answering the analytical question (to make sure that the generated session does not diverge from its goal) and coverage (number of time a recommendation is suggested).

3. Finally, to understand the overall benefit of Falseto assuming that the recommender system is used in combination with the log search facility, we use Falseto's recommender system on the sessions of the first generation and we report the metrics listed above.

5.2 Lessons learned

We begin by describing the log obtained. Overall, a log of 27 sessions (308 queries, 108 of them relevant) was collected. A analysis of the overall log is reported in Table 1 (effort is the inverse ratio of relevant queries per session, i.e., the number of queries it takes to obtain one relevant query).

We also characterized the sessions in terms of the OLAP operations used to edit the queries...

Analysis of the log	min	max	avg	std dev.
Size of session	5	31	11.4	6.35
Relevant queries / session	2	11	4	2.61
Ratio of relevant queries	0.12	0.71	0.37	0.19
Effort	1.4	8	3.08	1.76

Table 1: Analysis of the log

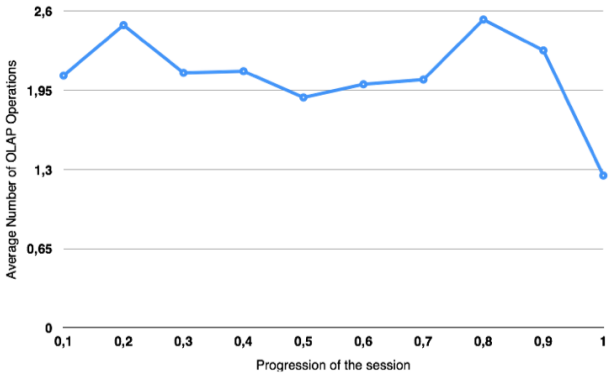

Figure 5: Number of OLAP operations between two consecutive queries

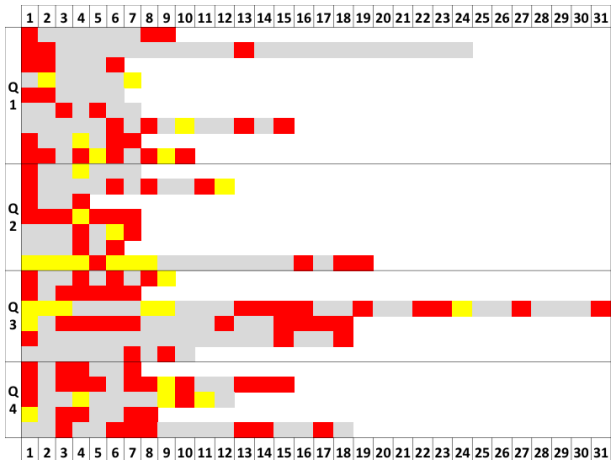

Figure 6: Distribution of relevant queries

Figure 5 shows the distribution of the number of OLAP operations used between two consecutive queries, along the sessions. On the average this number varies between 1.3 and 2.6 operations. Two peaks are observed, at the beginning of the session (around 20%) and near the end of it, before a final drop (for the last 10% of the session).

Figure 6 reports the distribution of the position of relevant queries (in red) in the sessions. Note that given the relative small size of the log, manual inspection was possible and done, and the queries in yellow color (around 9%) indicate the presence of a query that does not seem to contribute to the session (being a repetition of another query of the session, or comporting edition mistakes like forgeting to incorporate the good measure attribute to analyze).

An analysis by question is reported in table 2. We have characterized the average similarity of the sessions in the log, using the similarity defined in [5], which is quite low (0.14) as expected, as well as the intra-question similarity and the inter-question similarity. It can be noted that intra-question similarity shows important variations, which indicates that sessions developed for answering a given analytical question can be quite different. However, this intra-question similarity is almost always greater than the average session similarity of the complete log. A notable exception is question 3, where the intra-question similarity is lower than the average similarity to the sessions of both questions 2 and 4. Interestingly this is the question that generated the highest number of queries per sessions, which can be understood as the difficulty of the question.

We then report the results of the naive automatic session generation algorithms described above. For each of the questions and each of the tests, 10 sessions were generated. The result of the first test, aiming at understanding Falseto's log search capacity, is reported in table 3, that shows recall (R) and precision (P) for various average size of sessions (in brackets). We observe that

Question analysis	Q1	Q2	Q3	Q4
Number of sessions	8	7	6	5
Number of queries	82	57	86	56
Number of relevant queries	26	21	34	27
Queries / session	10.25	8.14	14.33	11.2
Relevant queries / session	3.25	3	5.67	5.4
Intra-question similarity	0.54	0.16	0.13	0.21
Inter-question similarity				
Q1	**0.54**			
Q2	0.02	**0.16**		
Q3	0.11	*0.15*	0.13	
Q4	0.14	0.08	*0.16*	**0.21**

Table 2: Question analysis

Log search	R(4)	P(4)	R(6)	P(6)	R(12)	P(12)
Q1	0.06	0.76	0.15	0.84	0.23	0.99
Q2	0.08	1	0.10	1	0.25	0.80
Q3	0.04	1	0.07	0.89	0.07	0.49
Q4	0.03	0.56	0.07	0.68	0.07	0.49

Table 3: Log search capacity

precision is almost always good (the log search facilities enables to find past queries developed for the question) while recall increases the average session size. A similar phenomenon occur when computing recall and precision with respect to relevant queries (not reported here due to lack of space).

This test shows how effective Falseto can be for retrieving in a log queries and therefore sessions developed for an analytical need that is the same of the one currently investigated by the current session.

The result of the second test, aiming at understanding the capacity of Falseto's recommener system, is reported in Table 4. It can be seen that coverage is very good (a recommendation is always proposed). Novelty is also very good in all the cases except for question 1, which can be explained by the fact

Recommender system	Q1	Q2	Q3	Q4
Average novelty	0	1	1	1
Average coverage	1	1	1	1
Average recommendation similarity				
Recommendations for Q1	**0.75**	0.05	0.19	0.20
Recommendations for Q2	0.02	**0.39**	0.32	0.29
Recommendations for Q3	0.08	0.19	**0.29**	0.21
Recommendations for Q4	0.006	0.18	0.26	**0.44**

Table 4: Recommender system capacity

Search and recommendations	Q1	Q2	Q3	Q4
Average recall	0.31	0.22	0.07	0.07
Average precision	0.95	0.74	0.53	0.36
Average novelty	0.2	0.8	0.7	0.9
Average coverage	1	1	1	1
Average recommendation similarity				
Recommendations for Q1	**0.61**	0.02	0.16	0.15
Recommendations for Q2	0.03	**0.38**	0.2	0.19
Recommendations for Q3	0.07	0.29	*0.25*	0.22
Recommendations for Q4	0.05	0.25	0.26	**0.36**

Table 5: Combining search facilities with recommendations

that the sessions developed for this question are very close to each other. Importantly, the recommendations are always closer to the sessions of the question compared to the sessions of other questions, which means that the focus on the question is preserved. This is further confirmed by the fact that the average similarity of the recommandations to the sessions of the question is even above the intra-question similarity.

The results of the last test (generating sessions by recommending for the sessions generated for test number 1) is reported in table 5. These results are slightly degraded compared to the previous tests, which is due to the combination of the two holistic functionalities. Overall, they confirm that, even with a very naive way of using Falseto's functionalities, the sessions obtained are focused on the analytical question addressed and include both previously developed and novel queries.

Efficiency Finally, we tested the responsiveness of Falseto, to check if the holistic functionalities are compatible with a interactive use. The tests were run on ???

Figure 7 shows the average time it takes to compute i) the initial summary using clustering, ii) a search in the log and iii) a recommendation, for logs of increasing sizes. It can be seen that log search and recommendation times are perfectly compatible with an interactive use (the maximum average was 152 ms in our tests, for computing recommendations with a log of 200 sessions). The time to compute the initial clustering is much more substantial and increases with the size of the log, but it remains acceptable since it is executed only once when Falseto is started. This test confirms Falseto's user-friendliness, and, importantly, justifies the use of only the query syntax (and not the query answer, as explained in Section 4) in the holistic functionalities of Falseto.

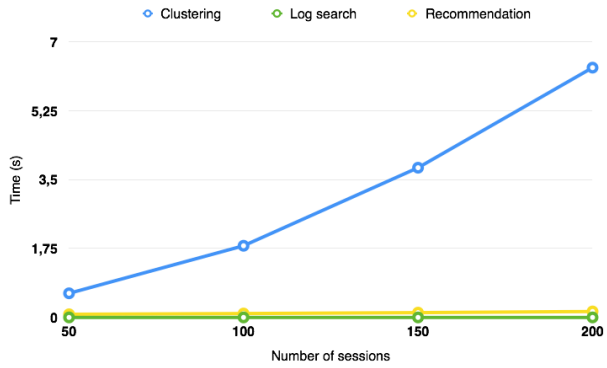

Figure 7: Falseto's responsiveness

6 Bridging the gap between OLAP and exploratory search

6.1 Related work

It is now admitted in the database community that query logs can be leveraged not only for physical tuning, but also for user empowerment [14]. Logs can indeed be used for learning user preferences [12], for query recommendation [7, 21], for query auto-completion [15], or for query composition [16]. In particular, the study of [16] showed that SQL query composition time can be heavily reduced when users are provided with a tool to browse, search and reuse former SQL queries organized in sessions.

In the OLAP context, the navigational nature of analytical sessions over data cubes is well known [19], and approaches have been developped for improving query performance holistically at the session level [19, 13], or suggesting queries or facts to assist the user in her navigation [20, 9]. Noticeably, while many efforts have been devoted to speedup OLAP query processing, the question of how effective an OLAP session is has attracted little attention, while it is predominent in other field like information retrieval or even more specifically, exploratory search.

Exploratory search [22] can be defined as a search paradigm centered on the user and the evolution of her knowledge, aiming at a better support for information understanding by moving beyond the traditional query-browse-refine model. As illustrated by Figure 8, exploratory search consists of two main activities: exploratory browsing and focused searching. Exploratory browsing essentially helps relate the problem context to similar documented experiences, while focused searching is generally intended to help the user follow a known

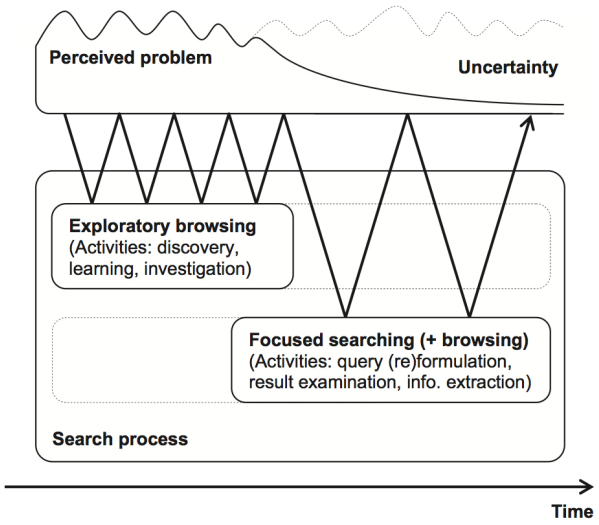

Figure 8: A model of exploratory search behavior (from [22])

or expected trail. Exploratory search is particularly driven by the quality of the user's experience, and suitable metrics for measuring it have been proposed [22]: *engagement and enjoyment* (the number of actionable events), *information novelty* (the rate at which new information is encountered), *task success* (do users reach a particular target and got enough detail en route to their goal), *task time* (the time spent to complete the task), and *learning and cognition* (the evolution of individual knowledge).

Exploratory search attracted the interest of the database community, and was even the topic of a recent post on the Sigmod blog (http://wp.sigmod.org/), with a specific section that compares it to OLAP. The conclusion of this section was that OLAP has serious limitations in supporting exploratory search, especially in the domain of discovery, adaptability to new information needs, and the support of all types of relevant data and users.

We next discuss how Falseto could help to bridge the gap between OLAP and exploratory search.

6.2 Bridging the gap

Figure 8 can be used as a first driver for bridging the gap between OLAP and exploratory search. The two languages implemented by Falseto and presented in Section 4 can be seen as targeting the two activities of exploratory search. On the one hand, the query edition language can be used for focused searching, and on the other hand, the log manipulation language can be used to describe exploratory browsing tasks that take advantage of former sessions.

With this idea in mind, we have designed a simple protocol to understand,

via user tests, how Falseto can be used to better support exploratory search in the context of OLAP. This protocol relies on the production of sessions in reponse to a given analytical question, and the production of reports by the inclusion and comment of relevant queries from the session in a document.

We first start by adapting to the OLAP context the metrics introduced above for assessing exploratory search applications. In the OLAP context, these metrics are computed as follows:

- User engagement is measured as the number of queries found relevant and included in the report created while answering a given analytical question. It is also measured as the number of clicks on the holistic functionalities.
- Information novelty is measured as the number of queries in a report that are relevant and that appear in no other reports of the same analytical need.
- The task time is measured as the time taken to complete a session (to answer the analytical question). The time spent on holistic functionalities (exploratory browsing) is also reported. We also measure the user effort as the ratio between the number of queries in the session and the number of queries in the report.
- The task success is measured as the precision and recall of the report that is the answer to a given analytical question. Precision is the number of relevant queries in a report over the overall number of queries in the report. Recall is the number of queries of a report that appear in (or that are included in a query of) another report.
- Finally, learning and cognition can be measured by recording a user profile and assess i) how this profile evolves through the consecutive uses of Falseto and ii) if this user profile leads to improvements of the other metrics through the consecutive uses of Falseto.

We now sketch the protocol more precisely. The complete protocol can be downloaded from the Falseto website ... It corresponds to a test that is organized in 3 phases. The first phase aims at giving the analysts the sufficient knowledge to use both the basic and holistic functionalities of Falseto. The analysts are asked to complete the Falseto tutorial. The tutorial explains Falseto's interface and functionalities, and proposes simple exercises that let the analysts develop queries and analytical sessions over the IPUMS census data cube [18]. After this phase, the test supervisors should constitute 3 groups of students (named Basic, Advanced and Complete from now on), ensuring in each group an even distribution of skills with Falseto.

For the second phase, the analysts are split into 3 groups of similar size: in the Basic group, the analysts will use the basic version of Falseto that includes the query and session designer and does not include the advanced functionalities (recommender system, log browsing and filtering). In the Advanced group, the analysts will use the advanced version of Falseto that only includes the advanced

functionalities (recommender system, log browsing and filtering), but does not allow the analyst to edit queries. In the Complete group, the analysts will use the complete version of Falseto that includes all the functionalities. All analysts are asked to solve an exercise consisting of 2 analytical needs expressed as questions that can be solved by navigating the census data cube. The second need is deliberately more fuzzy than the first one, and probably requires a longer navigation to get answered. The analysts should answer one requirement after the other. To answer the need, they must use the version of Falseto that corresponds to their group. The form of the answer to the need is twofold: the log that records the session (automatically generated by Falseto) and a report. This report is a document that contains a list of annotated query results, taken from the session.

During the last phase, each analyst should evaluate all the reports that were produced to answer a particular analytical need. They should indicate, for each of the queries if the query is relevant (does it contribute to answer the need, according to the analyst) and mark all the reports that include this query. If a report does not include the query stricto sensu but a superset of it (for instance the query filters out one year but the report includes a query that includes all the years but which apart from that is exactly the same) this report should also be marked as including the query.

Implementing this protocol is part of our future works. As mentioned above, OLAP is limited in its exploratory search capabilities in terms of discovery, adaptability to new information needs, and the support of all types of relevant data and users. Implementing this protocol will at least enable to answer to the limitations in terms of discovery (can Falseto leads the user towards interesting findings they would have missed without Falseto) and in term of supporting all type of users (naive and expert).

7 Conclusion

This paper introduced Falseto (Former AnalyticaL Sessions for lEss Tedious OLAP), a graphical OLAP tool that is meant to assist OLAP query and session composition. During a session with Falseto, the user can summarize, browse, query and reuse former sessions recorded in a query log, and take advantage of sessions suggested by Falseto's recommender system. This tool relies on a logical framework the includes a language for editing queries and a language for manipulating logs. We report the lessons learned from analyzing real OLAP sessions and using them to assess Falseto's functionalities. Finally we discussed how Falesto can link OLAP to the field of exploratory search, a search paradigm centered on the evolution of the user's understanding.

Falseto can be seen as the starting point of a platform for developing user-centric OLAP facilities. The first short term perspective of this work is to better assess Falseto's holistic functionalities with user tests, so as to improve the existing functionalities and devise new ones. Another short term perspective is to find better ways of graphically displaying sessions and summaries. On the

long run, our aim is to be able to benchmark user-centric OLAP activities, which would be possible by bridging the gap between OLAP and exploratory search.

References

[1] Rakesh Agrawal, Ashish Gupta, and Sunita Sarawagi. Modeling Multidimensional Databases. In *ICDE*, pages 232–243, 1997.

[2] Julien Aligon. *Similarity based recommendation of OLAP sessions*. PhD thesis, Université François Rabelais Tours, 2013.

[3] Julien Aligon, Enrico Gallinucci, Matteo Golfarelli, Patrick Marcel, and Stefano Rizzi. A collaborative filtering approach for recommending olap sessions. *Under submission*, 2014.

[4] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. Mining Preferences from OLAP Query Logs for Proactive Personalization. In *ADBIS*, pages 84–97, 2011.

[5] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. Similarity measures for olap sessions. *Knowl. Inf. Syst.*, 39(2):463–489, 2014.

[6] Julien Aligon, Haoyuan Li, Patrick Marcel, and Arnaud Soulet. Towards a logical framework for OLAP query log manipulation. In *PersDB 2012, 6th International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases*, 2012.

[7] Gloria Chatzopoulou, Magdalini Eirinaki, Suju Koshy, Sarika Mittal, Neoklis Polyzotis, and Jothi Swarubini Vindhiya Varman. The QueRIE system for Personalized Query Recommendations. *IEEE Data Eng. Bull.*, 34(2):55–60, 2011.

[8] Arnaud Giacometti, Patrick Marcel, and Elsa Negre. A framework for recommending OLAP queries. In *DOLAP*, pages 73–80, 2008.

[9] Arnaud Giacometti, Patrick Marcel, and Elsa Negre. Recommending multidimensional queries. In *DaWaK*, pages 453–466, 2009.

[10] Matteo Golfarelli and Stefano Rizzi. *Data Warehouse Design: Modern Principles and Methodologies*. McGraw-Hill, 2009.

[11] Jiawei Han, Yixin Chen, Guozhu Dong, Jian Pei, Benjamin W. Wah, Jianyong Wang, and Y. Dora Cai. Stream cube: An architecture for multidimensional analysis of data streams. *Distributed and Parallel Databases*, 18(2):173–197, 2005.

[12] Stefan Holland, Martin Ester, and Werner Kießling. Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications. In *PKDD*, pages 204–216, 2003.

[13] Niranjan Kamat, Prasanth Jayachandran, Karthik Tunga, and Arnab Nandi. Distributed and interactive cube exploration. In *ICDE*, pages 472–483, 2014.

[14] Nodira Khoussainova, Magdalena Balazinska, Wolfgang Gatterbauer, YongChul Kwon, and Dan Suciu. A Case for A Collaborative Query Management System. In *CIDR*, 2009.

[15] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. SnipSuggest: Context-Aware Autocompletion for SQL. *PVLDB*, 4(1):22–33, 2010.

[16] Nodira Khoussainova, YongChul Kwon, Wei-Ting Liao, Magdalena Balazinska, Wolfgang Gatterbauer, and Dan Suciu. Session-Based Browsing for More Effective Query Reuse. In *SSDBM*, pages 583–585, 2011.

[17] Ralph Kimball. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. John Wiley, 1996.

[18] Minnesota Population Center. Integrated public use microdata series. http://www.ipums.org, 2008.

[19] Carsten Sapia. PROMISE: Predicting Query Behavior to Enable Predictive Caching Strategies for OLAP Systems. In *DAWAK*, pages 224–233, 2000.

[20] Sunita Sarawagi. User-Adaptive Exploration of Multidimensional Data. In *VLDB*, pages 307–316, 2000.

[21] K. Stefanidis, M. Drosou, and E. Pitoura. "You May Also Like" results in relational databases. In *Proceedings International Workshop on Personalized Access, Profile Management and Context Awareness: Databases*, Lyon, France, 2009.

[22] Ryen W. White and Resa A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2009.